```
**********************************************************
   41282 Wed Jan 30 11:28:01 2019
new/usr/src/common/smbios/smb_info.c
10145 smbios_info_boot() gets NULL check wrong
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2015 OmniTI Computer Consulting, Inc.  All rights reserved.
 * Copyright (c) 2018, Joyent, Inc.
 * Copyright (c) 2017, Joyent, Inc.
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * SMBIOS Information Routines
 *
 * The routines in this file are used to convert from the SMBIOS data format to
 * a more reasonable and stable set of structures offered as part of our ABI.
 * These functions take the general form:
 *
 *      stp = smb_lookup_type(shp, foo);
 *      smb_foo_t foo;
 *
 *      smb_info_bcopy(stp->smbst_hdr, &foo, sizeof (foo));
 *      bzero(caller's struct);
 *
 *      copy/convert foo members into caller's struct
 *
 * We copy the internal structure on to an automatic variable so as to avoid
 * checks everywhere for structures that the BIOS has improperly truncated, and
 * also to automatically handle the case of a structure that has been extended.
 * When necessary, this code can use smb_gteq() to determine whether the SMBIOS
 * data is of a particular revision that is supposed to contain a new field.
 *
 * Note, when trying to bzero the caller's struct you have to be careful about
 * versions. One can only bzero the initial version that existed in illumos. In
 * other words, if someone passes an older library handle that doesn't support a
 * version you cannot assume that their structures have those additional members
 * in them. Instead, a 'base' version is introduced for such types that have
 * differences and instead we only bzero out the base version and then handle
 * the additional members. In general, because all additional members will be
 * assigned, there's no reason to zero them out unless they are arrays that
 * won't be entirely filled in.
 *
 * Due to history, anything added after the update from version 2.4, in other
```

```
 * words additions from or after '5094 Update libsmbios with recent items'
 * (4e901881) is currently being used for this. While we don't allow software
 * compiling against this to get an older form, this was the first major update
 * and a good starting point for us to enforce this behavior which is useful for
 * moving forward to making this more public.
 */

#include <sys/smbios_impl.h>
#include <sys/byteorder.h>

#ifdef _KERNEL
#include <sys/sunddi.h>
#else
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#endif

/*
 * A large number of SMBIOS structures contain a set of common strings used to
 * describe a h/w component's serial number, manufacturer, etc.  These fields
 * helpfully have different names and offsets and sometimes aren't consistent.
 * To simplify life for our clients, we factor these common things out into
 * smbios_info_t, which can be retrieved for any structure.  The following
 * table describes the mapping from a given structure to the smbios_info_t.
 * Multiple SMBIOS stuctures' contained objects are also handled here.
 */
static const struct smb_infospec {
        uint8_t is_type;                /* structure type */
        uint8_t is_manu;                /* manufacturer offset */
        uint8_t is_product;             /* product name offset */
        uint8_t is_version;             /* version offset */
        uint8_t is_serial;              /* serial number offset */
        uint8_t is_asset;               /* asset tag offset */
        uint8_t is_location;            /* location string offset */
        uint8_t is_part;                /* part number offset */
        uint8_t is_contc;               /* contained count */
        uint8_t is_contsz;              /* contained size */
        uint8_t is_contv;               /* contained objects */
} _smb_infospecs[] = {
______unchanged_portion_omitted_

id_t
smbios_info_boot(smbios_hdl_t *shp, smbios_boot_t *bp)
{
        const smb_struct_t *stp = smb_lookup_type(shp, SMB_TYPE_BOOT);
        const smb_boot_t *b;
        const smb_boot_t *b = (smb_boot_t *)(uintptr_t)stp->smbst_hdr;

        if (stp == NULL)
                return (-1); /* errno is set for us */

        bzero(bp, sizeof (smbios_boot_t));

        b = (smb_boot_t *)(uintptr_t)stp->smbst_hdr;

        bp->smbt_status = b->smbbo_status[0];
        bp->smbt_size = stp->smbst_hdr->smbh_len - sizeof (smb_boot_t);
        bp->smbt_data = bp->smbt_size ? &b->smbbo_status[1] : NULL;

        return (stp->smbst_hdr->smbh_hdl);
}
______unchanged_portion_omitted_
```